1

First Post

WHAT'S IN THIS CHAPTER?

- Appreciating the provenance of the WordPress platform
- Choosing a suitable platform for your WordPress installation
- Downloading, installing, and performing basic configuration of WordPress
- Diagnosing and resolving common installation problems

If displaying "Hello World" on an appropriate device defines minimum competence in a programming language, generating your first post is the equivalent in the blogging world. This chapter provides a brief history of WordPress and then explores several options for hosting a WordPress installation. Common miscues and misperceptions along with their resolutions round out the chapter and put you on the edge of publishing your wit and wisdom.

Once you've installed, configured, and completed the bare-bones administration, you're ready to take advantage of the code walkthroughs and detailed component descriptions in later chapters. Of course, if you already have a functional WordPress blog, you can skip this chapter and dive head-first into the Dashboard control wonderland in Chapter 2, "Functional Overview."

COPYRIGHTED MATERIAL

WHAT IS WORDPRESS?

WordPress is one of the most popular open source blogging systems available, with global and vibrant user, developer, and support communities. Though it can be compared to TypePad, Moveable Type, Google's Blogger, and the Apache Roller project as a user-generated content workhorse, WordPress distinguishes itself with a broad array of hosting options, functional extensions (plugins), and aesthetic designs and elements (themes).

With the rise of self-publishing, low-cost web hosting and freely available core components like the MySQL database, blogging software followed the same trend as most other digital technologies, moving from high-end, high-cost products to widely available, low-cost consumer or "hobbyist" systems. WordPress isn't simply about creating a blog so that you can have

a digital diary attached to your vanity URL; it has evolved into a full-fledged content management system used by individuals and enterprises alike. This section takes a brief tour through the early history of WordPress and brings you up to speed on the current release and user community.

A Short History: WordPress and Friends

WordPress started similarly to many other popular open source software packages: Some talented developers saw a need to create a powerful, simple tool based on an existing project licensed under the GPL. Michel Valdrighi's b2/cafelog system provided the starting point, and WordPress was built as a fork of that base by developers Matt Mullenweg and Mike Little. WordPress first appeared in 2003, also built on the MySQL open source database for persisting content and PHP as the development platform. Valdrighi remains a contributor to the project, which thrives and depends on a growing and interested community of users and developers.

As with other systems written in PHP, it is self-contained in the sense that installation, configuration, operation, and administration tasks are all contained in PHP modules. WordPress's popularity has been driven in part by its simplicity, with the phrase "five minute installation" making appearances in nearly every description or book about WordPress. Beyond getting to a first post, WordPress was designed to be extended.

WordPress today is supported by a handful of core developers and just under 100 key contributors. Mike Little today runs the WordPress specialty shop `zed1.com` and he contributes the occasional patch to the code. Matt Mullenweg's company, Automattic, continues to operate the `wordpress.com` hosting service as well as fund development of related content management tools: WordPress MU, a multi-user version of WordPress that is at the heart of the `wordpress.com` hosting system. Pronounce it "em-you" or take the rather scholarly "myu" approach if you want to impress your Greek or mathematically inclined friends. Gravatar dynamically serves images tied to e-mail addresses, providing a hosted icon with a variety of display options. Think of it as a service to make hot-linking your profile picture technically and socially acceptable.

As a content management system, the WordPress system definition doesn't stop at time-serialized posts with comments. BuddyPress is a set of themes and plugins that extends WordPress into a functional social networking platform, allowing registered users to message and interact with each other, again with all content managed within the WordPress framework. Similarly, bbPress is a PHP- and MySQL-based system designed for forums (bulletin boards) that is distinct from WordPress but is commonly integrated with it.

We cover some of these WordPress adjunct systems in more detail in Chapter 15, "The WordPress Developer Community," but they're included here to provide a sense of how WordPress has expanded beyond a basic single-user-oriented tool. At the same time, we're not endorsing or making a commercial for Automattic, but delving into the guts of WordPress without a spin of the propeller hat toward Mullenweg and Little is somewhere between incorrigible and bad community behavior.

Current State

This book is based on the WordPress 2.9 major release. Each successive release of WordPress has included improvements in the administration and control functions (Dashboard), backup, export, and import functions, and installation and upgrade features. Even if you start with a slightly down-rev

version of WordPress, you'll be able to bring it up to the current release and maintain the freshness of your install. We touch on install and upgrade paths later in this chapter.

Exactly how popular and prevalent is WordPress usage? "Popular" is always a subjective metric, but statistics add some weight to those perceptions. Jason Calacanis claimed 202 million websites using WordPress in Episode 16 of "This Week in Startups" (September 2009). That includes sites using WordPress for content management, blogging, and personal rants, and has to be discounted by those of us who have multiple WordPress installations to their names, but even with that order of magnitude estimate, WordPress is immensely popular.

Here are download statistics for the core WordPress system:

- **2006:** 1.5 million (source: WordPress.org)
- **2007:** 3.8 million (source: WordPress.org)
- **2008:** More than 11 million (source: Matt Mullenweg's WordCamp NYC keynote)

Hosted blogs on `wordpress.com` now number over 4.6 million, with more than 35 million posts over the 2008 calendar year, reaching a run rate of about 4 million posts per month; again those statistics are courtesy of Mullenweg's WordCamp NYC keynote (available on WordCampTV for your viewing pleasure). The plugin population went from about 370 in 2006, to 1,384 in 2007, with more than 6,300 currently registered as reported on WordPress.org. The combinations of plugins and themes require scientific notation to represent in complexity, but at the same time, they're all equally simple to locate, integrate, and use. That's the result of a solid architecture and an equally solid community using it.

Today, WordPress powers CNN's blogs, the *Wall Street Journal*'s *All Things D*, and the irreverent but snowclone-driven `icanhazcheeseburger.com`. (If you looked for a backstory on "snowclone," apologies, but that's also the joy of discovering new facts in a culture of participatory media).

Where do you get started?

`wordpress.org` is the home for the current released and in-development versions of the code. Click down to `wordpress.org/extend` for a starting point in finding plugins, themes, and wish lists of ideas and features to be implemented.

`wordpress.com` has both free and paid hosting services. Over at `wordpress.org/hosting` you'll find a list of hosting providers that support WordPress and often include some additional first-time installation and configuration support in their packaging of the code for delivery as part of their hosting services.

Intersecting the Community

WordPress thrives and grows based on community contributions in addition to sheer usage. Like high school gym class, participation is the name of the game, and several semi-formal avenues along which to channel your efforts and energies are available.

WordCamp events are community-hosted, locally operated, and now happen in dozens of cities around the world. Camps that reach critical mass are listed on wordcamp.org, but you'll do just as well to search for a WordCamp event in a major city close to you. WordCamps occur nearly every weekend with bloggers, photographers, writers, editors, developers, and designers of all experience and skill levels counted among their attendees. WordCamps are a low-cost introduction to the local community and often a good opportunity to meet WordPress celebrities.

Less structured but more frequently convened than WordCamps are WordPress Meetups, comprising local users and developers in more than 40 cities. You'll need a `meetup.com` account, but once you're registered you can check on locations and timetables at `wordpress.meetup.com` to see when and where people are talking about content management.

A rich, multi-language documentation repository is hosted at `codex.wordpress.org`. The WordPress Codex, with all due respect to the term reserved for ancient handwritten manuscripts, represents the community-contributed tips and tricks for every facet of WordPress from installation to debugging. It's a wiki with fourteen administrators and well over 70,000 registered users. If you feel the urge to contribute to the WordPress documentation, register and write away in the WordPress Codex. We hope you'll find this book a cross between a companion and a travel guide to the Codex.

Finally, mailing lists (and their archives) exist for various WordPress contributors and communities. A current roster is available online at `codex.wordpress.org/Mailing_Lists`; of particular interest may be the wp-docs list for Codex contributors and the wp-hackers list for those who work on the WordPress core and steer its future directions.

WordPress and the GPL

WordPress is licensed under the Gnu Public License (GPL) version 2, contained in the `license.txt` file that you'll find in the top-level code distribution. Most people don't read the license, and simply understand that WordPress is an open source project; however, pockets of corporate legal departments still worry about the viral component of a GPL license and its implications for additional code or content that gets added to, used with, or layered on top of the original distribution. Much of this confusion stems from liberal use of the words "free" and "copyright" in contexts where they are inappropriately applied.

We're not lawyers, nor do we play them on the Internet or on television, and if you really want to understand the nuances of copyright law and what constitutes a "conveyance" of code, pick up some of Lawrence Lessig's or Cory Doctorow's work in those areas. We include this section to assuage IT departments who may be dissuaded from using WordPress as an enterprise content management system by overly zealous legal teams. Don't let this happen to you; again, if WordPress is acceptable to CNN and the *Wall Street Journal*, two companies that survive on the copyrights granted to their content, it probably fits within the legal strictures of most corporate users as well.

The core tenet of the GPL ensures that you can always get the source code for any distribution of GPL-licensed software. If a company modifies a GPL-licensed software package and then redistributes that newer version, it has to make the source code available as well. This is the "viral" nature of GPL at work; its goal is to make sure that access to the software and its derivatives is never reduced in scope. If you plan on modifying the WordPress core and then distributing that code, you'll need to make sure your changes are covered by the GPL and that the code is available in source code form. Given that WordPress is written in PHP, an interpreted language, distributing the software and distributing the source code are effectively the same thing.

Following are some common misperceptions and associated explanations about using WordPress in commercial situations.

"Free software" means we can't commercialize its use. You can charge people to use your installation of WordPress, or make money from advertisements running in your blog, or use a WordPress content

management platform as the foundation of an online store. That's how `wordpress.com` works; it also enables Google to charge advertisers for using their Linux-based services. You can find professional quality WordPress themes with non-trivial price tags, or you can pay a hosting provider hundreds or thousands of dollars a year to run your MySQL, PHP, Apache, and WordPress software stack; both involve commercialization of WordPress.

If we customize the code to handle our own {content types, security policies, obscure navigational requirements} we'll have to publish those changes. You're only required to make the source code available for software that you distribute. If you choose to make those changes inside your company, you don't have to redistribute them. On the other hand, if you've made some improvements to the WordPress core, the entire community would benefit from them. Getting more staid employers to understand the value of community contribution and relax copyright and employee contribution rules is sometimes a bit challenging, but the fact that you had a solid starting point is proof that other employers made precisely that set of choices on behalf of the greater WordPress community.

The GPL will "infect" content that we put into WordPress. Content — including graphical elements of themes, posts, and pages managed by WordPress — is separated out from the WordPress core. It's managed by the software, but not a derivative of or part of the software. Themes, however, are a derivative of the WordPress code and therefore also fall under the GPL, requiring you to make the source code for the theme available. Note that you can still charge for the theme if you want to make it commercially available. Again, the key point here is that you make the source code available to anyone who uses the software. If you're going to charge for the use of a theme, you need to make the source code available under the GPL as well, but as pointed out previously, users installing the theme effectively get the source code.

More important than a WordPress history lesson and licensing examination are the issues of what you can do with WordPress and why you'd want to enjoy its robustness. The next section looks at WordPress as a full-fledged content management system, rather than simply a blog editing tool.

CONTENT AND CONVERSATION

Multiple linear feet of shelves in bookstores are filled with volumes that will improve your writing voice, literary style, blogging techniques, and other aspects of your content creation abilities. One of our goals for this book is define the visual, stylistic, and context management mechanisms you can build with WordPress to shape vibrant user communities around your content. That context stimulates conversation with your readers. It's not just about the words in each post, or even if you're an interesting writer. How will people find you? How will you stand out in the crowd? How do you put your own imprint on your site, and personalize it for whatever purpose: personal, enterprise, community, or commercially measured?

WordPress as a Content Management System

Blogging systems have their roots in simple content management operations: create a post, persist it in stable storage such as a filesystem or database, and display the formatted output based on some set of temporal or keyword criteria. As the richness and types of content presented in blog pages expanded, and the requirements for sorting, searching, selecting, and presenting content grew to include

metadata and content taxonomies, the line between vanilla, single-user–targeted blogging software and enterprise-grade content management systems blurred.

Content management systems (CMS) handle the creation, storage, retrieval, description or annotation, and publication or display of a variety of content types. CMS also covers workflow tasks, typically from an editorial or publishing perspective, but equally including actions such as approval and marking content for additional editing or review. The WordPress Dashboard, covered in detail in Chapter 2, provides those elements of workflow management and editorial control. WordPress isn't the only open source content management system in widespread use today; the Drupal and Joomla projects are equally popular choices. Drupal and Joomla start from the perspective of managing content repositories; they handle a variety of content types, multiple authors in multiple roles, and getting the content to a consumer that requests it. WordPress is at its heart a blogging system, and the end focus is on displaying content to a reader. Although areas of functional overlap exist, you can integrate WordPress with other content management systems, a process covered in detail in Chapter 12.

WordPress has established itself as a *bona fide* content management system through its design for extensibility and the separation of content persistence from content display. Taking some liberties with the Model-View-Controller design pattern, WordPress separates the MySQL persistence layer as a data model, the theme-driven user interface and display functions, and the plugin architecture that interposes functionality into the data to presentation flow. Most important, WordPress stores content in raw form, as input by the user or an application posting through the WordPress APIs. Content is not formatted, run through templates, or laid out until the page is rendered, yielding immense power to the functions that generate the actual HTML. At the same time, the data model used by WordPress uses a rich set of tables to manage categories (taxonomies), content tags (folksonomies), author information, comments, and other pieces of cross-reference value. We explore the WordPress database schema that makes this possible in Chapter 6.

Although that design gives WordPress incredible power and flexibility as a content management system, it also requires knowledge of how those data persistence and control flows are related (it was a search for such a dissection of WordPress in functional terms that got us together to write this book).

Creating Conversation

> *Conversation is king; content is just something to talk about.*
>
> — Cory Doctorow

A robust CMS is measured by the utility of its content. Even the richest content types and most well-managed processes are of low return if nobody actually consumes the outputs. It's not sufficient to install blogging software, write a few posts, and hope the world shows up on your virtual doorstep; you need to create what Tim O'Reilly calls an "architecture of participation." Social networking, advertising, feeds, and taking steps to ensure your site shows up in search engine results will drive readers to your site; the design, branding, and graphic elements coupled with the quality of your content will encourage them to take the steps toward active participation.

Look at the problem from the perspective of a reader: in a world of tens of millions of blogs (many of which have a "first post" and not much else) how will you be found, heard, and echoed? Your Twitter followers should want to read your blog, and your WordPress blog can update your Twitter feed.

Conversely, your Twitter updates may appear in your WordPress blog's sidebar, marrying the ultra-short content timeline to the more thoughtful one. If you're active on Facebook, you can import blog entries into a public figure page (you're a writer, if not a more famous and self-promoting category), and Facebook readership will drive traffic back to your blog. If you cover specific, detailed, or arcane areas in your writing, Google searches for those terms should direct readers to your blog, where they'll join the conversation. We cover getting content into WordPress from social media and other content systems in Chapter 9, "Content Aggregation," and look at how your WordPress content can be more broadly distributed in Chapter 10, "Crafting a User Experience."

GETTING STARTED

Before any serious work on presentation, style, or content begins, you need a home for your blog (despite the previous discussion about WordPress and content management systems, we'll refer to your blog and the actual WordPress installation that implements it interchangeably, mostly for convenience and brevity). Factors affecting your choice include:

- **Cost:** Free hosting services limit your options as a developer, and frequently preclude you from generating money from advertising services. More expensive offerings may include better support, higher storage or bandwidth limits, or multiple database instances for additional applications.
- **Control:** What tools are provided for you to manage your MySQL database, files comprising the WordPress installation, and other content types? If you want to be able to muck around at the SQL level, or manage MySQL through a command-line interface, you should ensure your hosting provider supports those interfaces.
- **Complexity:** You can install the Apache web server with a PHP interpreter, MySQL, and the WordPress distribution yourself, but most hosting providers have wrapped up the installation process so that some of the rough edges are hidden from view. If you expect to need technical support on the underlying operating system platform, find a provider (including your own IT department) that provides that support in a reasonable time frame.

This section takes a quick look at some hosting options, walks through the basics of a do-it-yourself installation, and concludes with an overview of the ways in which WordPress and MySQL choose to ignore each other when installation goes into the weeds.

Hosting Options

Three broad categories of WordPress hosting exist, each with trade-offs between administrative complexity and depth of control. The easiest and most popular is to use `wordpress.com`, a free hosting service run by Automattic using WordPress MU. You can install themes and plugins through the Dashboard but you can only enable or disable the choices pre-installed for you. Further, you won't have access to the underlying MySQL databases, core code, or be able to integrate WordPress with other systems. You can redirect one of your own URLs to `wordpress.com`, but if you want full control over everything from the code to the URLs used, you're probably looking at a paid option. For our readers, the free route may be a reasonable first step, but we're assuming you're going to want to perform surgery on your installation.

You'll find a starter list of for-fee hosting providers on WordPress.org, including the paid option on `wordpress.com`. Most have the latest, or close to latest, releases of the WordPress core available as a package to be installed in conjunction with MySQL and a web server. The third hosting option is to install everything on servers that you own and operate. If your servers live in a hosting facility but you enjoy root administrative access, that's equivalent to a do-it-yourself installation.

WordPress requires a web server with PHP support, a URL rewriting facility, and an instance of MySQL. Apache is the most popular option for front-ending WordPress because it provides PHP interpretation through `mod_php` and URL rewriting in `mod_rewrite`. There is growing interest in lighttpd (Lighty) as a replacement for Apache, although the URL rewriting functionality needs a bit of handholding. Finally, you can use Microsoft's IIS 7.0 as a web server with its `URL_rewrite` module.The emphasis on URL rewriting stems from WordPress's support for "pretty" permalinks to blog entries, allowing you to create a URL tree organized by date, category, tag, or other metadata. Those mnemonic or readable URLs are mapped into MySQL database indices for the corresponding pages using the `.htaccess` file (in Apache parlance). It's a case of dynamic content generation, this time starting from the user path to the page in question, and relying heavily on your web server's muscle to map public neatness into internal structure. Technically, URL rewriting isn't required to install WordPress, but it's good to have because it gives you tremendous flexibility in the presentation and naming conventions used for your content's URLs. We cover permalink design and practices more in Chapter 2, but keep the requirement in mind as you select your WordPress substrate.

Up to this point we've mentioned MySQL in passing, but a brief review of MySQL requirements rounds out the hosting prerequisite list. It's worth establishing some terminology and distinguishing between the MySQL software, database instances, and WordPress instances using MySQL. When you install and configure MySQL, you have a full-fledged relational database system up and running. It doesn't have to be configured on the same machine as your web server, and some hosting providers will create horizontally scalable MySQL "farms" in parallel to their web server front ends. An *instance* of MySQL running on a server can support multiple *databases*, each with a unique name. When you install WordPress, you'll need to know the *name* of the MySQL database reserved for your content, although this information may be auto-generated and configured for you if you're using a provider that supports WordPress and MySQL as an integrated package. WordPress creates a number of relational data *tables* in that named database for each blog that you create.

Confusion results from nomenclature and complexity. You (or your hosting provider) may run multiple MySQL instances on multiple servers, and you'll need to know where your database is hosted. Because each instance of MySQL can run multiple databases, and each database contains groups of tables, it's possible to run multiple MySQL based applications on the same hosting platform, using one MySQL instance or even one MySQL database.

If you want to have multiple WordPress blogs on the same server, you can share a single MySQL database instance for all of them provided you configure WordPress to distinguish the *MySQL database table names* within the MySQL database. It's a simple configuration option that we cover in the next section, and it highlights the distinction between multiple sets of tables in a database and multiple databases for distinct applications.

Once you've secured the necessary foundation, it's time to get the code up and running. Even if you're using a hosting provider that installs MySQL and WordPress for you, it's worth knowing how the server-side components interact in case you need to track down a problem when you're deep in plugin development.

Do It Yourself Installation

The famous, fabled, fabulous five-minute WordPress installation is a reality when everything is configured and coordinated properly. This section walks through the steps that are often hidden from view when you use a provider with packaged installs, and highlights some of the common misfires between WordPress and MySQL instances.

The installation process is quite simple (assuming that your web server and MySQL server are already running): Download the WordPress package and install it in your web server's directory tree, then navigate to your top-level URL and complete the configuration. One (compound) sentence describes it completely.

It's possible and even advisable to install a fully functioning WordPress instance on your laptop or development machine, particularly if you are going to be working on the core, developing plugins or otherwise making changes that would create embarrassing failures during testing on a public web site. MacOS X comes with an Apache web server (with PHP and URL rewriting); download MySQL from `mysql.com`, or use a pre-packaged configuration like MAMP (`mamp.info`, which includes the phpMyAdmin tool) and you'll have a self-contained development and deployment lab. For other platforms, XAMPP (`www.apachefriends.org`) has a neatly integrated platform stack that runs on Windows, Mac OS and Linux foundations. Having everything under one hood is a powerful option for examining failure modes, as you'll see in the next two sections.

Installing WordPress Files

If you download the WordPress code from `wordpress.org`, you'll get a zip (or tarball) archive that expands into a directory called "wordpress." The first part of a WordPress installation is to get the code into your web server's directory structure, and ensuring you have it in the right place is a critical step. Gloss over this part and you'll find your blog ends up with a URL like `example.com/wordpress` and you'll either have to start over or e-mail ugly URLs to your friends and family. If that's what you want, to distinguish your blog from other content on your web site or to isolate multiple blogs, choosing the filesystem layout is equally important.

Pick the top-level directory where you want to install WordPress. Most commonly, this is the root directory for your web server, and if you're using a hosting provider it's probably the subdirectory called `public_html` in the file tree. If you are using a packaged install where there's a menu asking you for the target location, make sure you pick this top-level directory (and yes, you know that it already exists, that's the point!); if you're copying files from your local machine to the web server target using an FTP client, make sure you pick the right destination. The somewhat obvious move to copy the zip file to the server then unpack it will put everything into a "wordpress" subdirectory, and if you want your blog's URL to be `example.com` rather than `example.com/wordpress`, move the files "up" one directory level before proceeding. There is a configuration option to have your WordPress installation in a subdirectory to your top-level URL, so it's not fatal if you drop WordPress into a less-than-desirable filesystem geography. We cover that at the end of this section.

Once the WordPress files are installed, your filesystem browser should show you something like Figure 1-1, with an `index.php` and template `wp-config-sample.php` file. That's the entirety of the WordPress system, which runs effectively within the web server's PHP interpreter.

At this point, if you're doing a manual installation, you'll want to create your own `wp-config.php` file by editing the sample provided and saving it in your top-level WordPress directory. As an alternative, you can navigate to your blog's URL, and the WordPress code will realize there's no configuration file

and present you with a dialog boxes like those in Figures 1-2 and 1-3 where you can fill in the details. You'll need the MySQL database name, database username, and some idea of the WordPress database table prefix (other than the default wp_). These lower-level details are the guts of the next section on database configuration. If you are using a hosting provider with packaged installations you probably won't see this step, because the WordPress files will be extracted and the MySQL database information will be automatically inserted into a configuration file, no blogger-serviceable parts inside.

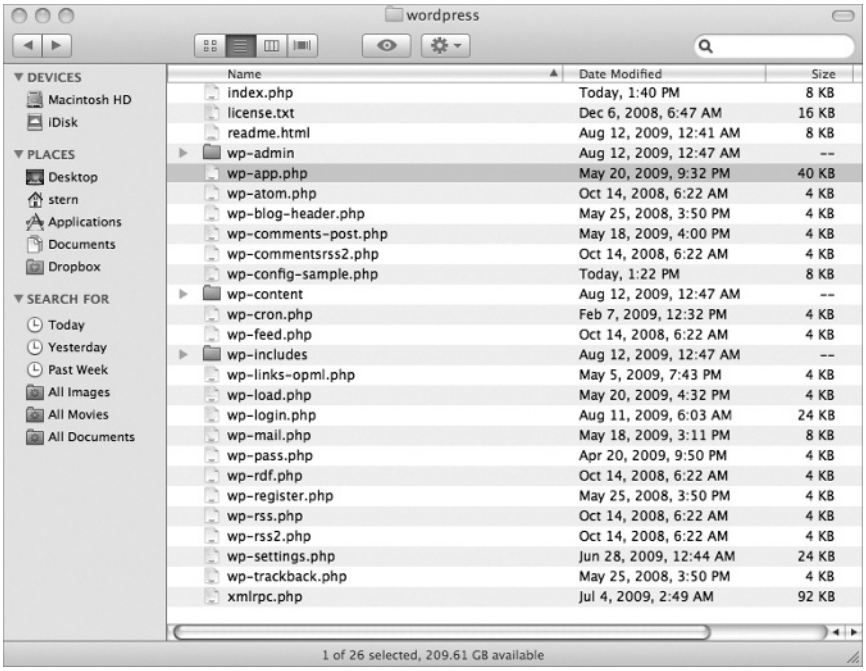
wordpress
DEVICES
Macintosh HD
iDisk
PLACES
Desktop
stern
Applications
Documents
Dropbox
SEARCH FOR
Today
Yesterday
Past Week
All Images
All Movies
All Documents
Name
Date Modified
Size
index.php Today, 1:40 PM 8 KB
license.txt Dec 6, 2008, 6:47 AM 16 KB
readme.html Aug 12, 2009, 12:41 AM 8 KB
wp-admin Aug 12, 2009, 12:47 AM --
wp-app.php May 20, 2009, 9:32 PM 40 KB
wp-atom.php Oct 14, 2008, 6:22 AM 4 KB
wp-blog-header.php May 25, 2008, 3:50 PM 4 KB
wp-comments-post.php May 18, 2009, 4:00 PM 4 KB
wp-commentsrss2.php Oct 14, 2008, 6:22 AM 4 KB
wp-config-sample.php Today, 1:22 PM 8 KB
wp-content Aug 12, 2009, 12:47 AM --
wp-cron.php Feb 7, 2009, 12:32 PM 4 KB
wp-feed.php Oct 14, 2008, 6:22 AM 4 KB
wp-includes Aug 12, 2009, 12:47 AM --
wp-links-opml.php May 5, 2009, 7:43 PM 4 KB
wp-load.php May 20, 2009, 4:32 PM 4 KB
wp-login.php Aug 11, 2009, 6:03 AM 24 KB
wp-mail.php May 18, 2009, 3:11 PM 8 KB
wp-pass.php Apr 20, 2009, 9:50 PM 4 KB
wp-rdf.php Oct 14, 2008, 6:22 AM 4 KB
wp-register.php May 25, 2008, 3:50 PM 4 KB
wp-rss.php Oct 14, 2008, 6:22 AM 4 KB
wp-rss2.php Oct 14, 2008, 6:22 AM 4 KB
wp-settings.php Jun 28, 2009, 12:44 AM 24 KB
wp-trackback.php May 25, 2008, 3:50 PM 4 KB
xmlrpc.php Jul 4, 2009, 2:49 AM 92 KB
1 of 26 selected, 209.61 GB available

FIGURE 1-1: A clean but unconfigured WordPress installation

What do you do if you already have HTML or other content at your target URL, and you want to add WordPress to an existing site? Disposition of existing files depends on your desired first user experience upon navigating to your URL. If you want visitors to see your blog, and to use WordPress as a content management system as we've described here, your best choice is to save existing content and convert it into blog posts or pages, effectively making your previous site color commentary and context for your WordPress-driven site. Alternatively, you can install WordPress in a subdirectory, keep your existing `index.html` file, and direct readers to your blog through a button or link on your extant home page. Don't leave this to chance; if you have an `index.html` file and then install WordPress, you'll have an `index.php` and an `index.html` file side by side and users will see one or the other depending upon the Directory Index configuration of your site's web server. Actions on existing content should be informed by how much traffic that content is driving to your site: if your pages are responsible for search engine traffic, you probably don't want to disrupt the existing URLs that have been cached, and should install WordPress in a subdirectory. If you feel strongly about making WordPress the wrapper around the user experience, move the content and include URL rewriting or redirection for pages that move into the WordPress world. We'll cover migrating existing content from a variety of formats and systems into WordPress in Chapter 14.

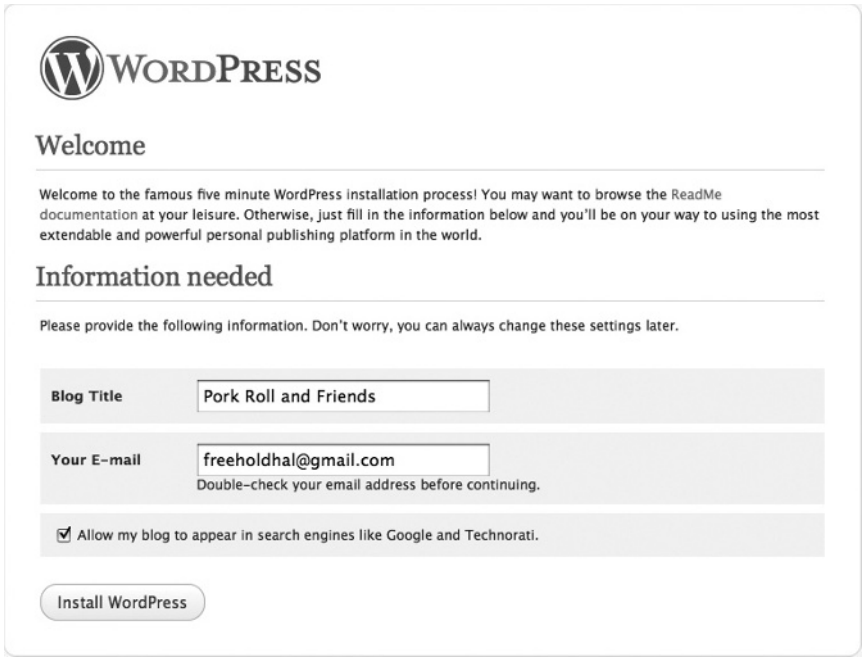
WordPress
Welcome
Welcome to the famous five minute WordPress installation process! You may want to browse the ReadMe documentation at your leisure. Otherwise, just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.
Information needed
Please provide the following information. Don't worry, you can always change these settings later.
Blog Title
Pork Roll and Friends
Your E-mail
freeholdhal@gmail.com
Double-check your email address before continuing.
Allow my blog to appear in search engines like Google and Technorati.
Install WordPress

FIGURE 1-2: Basic auto-configuration dialog box

WordPress
Below you should enter your database connection details. If you're not sure about these, contact your host.
Database Name
wp_halstern
The name of the database you want to run WP in.
User Name
wp_user1
Your MySQL username
Password
foobar
...and MySQL password.
Database Host
localhost
99% chance you won't need to change this value.
Table Prefix
wp_hs_
If you want to run multiple WordPress installations in a single database, change this.
Submit

FIGURE 1-3: Database configuration dialog box

If you used a hosting provider's packaged installation, or if you manually created a `wp-config.php` file and then navigated to your top-level blog URL, WordPress should have completed creating the database tables, created an administrative user for your blog, and set an initial password. Upon a successful installation, you should see a box like Figure 1-4 that indicates your five minutes of famed installation are done.

FIGURE 1-4: Administrative information at the conclusion of a clean install

The next section covers the MySQL-WordPress configuration dance in more detail, and is suitable reading even if thinking about SQL gives you hives. If you're up and running, you can skip the next section and go right into "First Time Administration."

Database Configuration

If your hosting provider spun up a MySQL database and created a user for you, check your resultant `wp-config.php` file to gather this information. It is necessary for the MySQL probing covered in this section, and it's good to have in case you run into MySQL problems later on. There's a username and password combination included in that file, so treat it the way you'd treat other login information. On the other hand, if you're going deep on the do-it-yourself route, this section gives you a sense of what's likely to create confusion or consternation as you pull the pieces together.

In theory, MySQL set up for WordPress is trivial: make sure MySQL is up and running, create a WordPress user in MySQL, and then have that user create a database to hold the WordPress tables. You can use the MySQL command line or tools like phpMyAdmin for these tasks, but bear in mind that MySQL has its own set of users and permissions granted to those users, distinct from those used by your (or your hosting provider's) operating system. Once MySQL is installed, it will create a default table of users and grants, adding a "root" user on Unix systems that is a MySQL superuser, unrelated to the Unix root user. However, if you're attempting to connect to your MySQL instance as the MySQL root user, those connections can only be made from localhost – the same machine on which MySQL is running. If you want to learn more about MySQL permissions, the table governing grants of those permissions to users, and how MySQL users are managed, refer to the "MySQL Reference Manual" (`http://dev.mysql.com/doc/`) and the sections on securing the initial MySQL accounts.

No set naming conventions exist for WordPress users or databases; hosting providers will typically prepend the name of the package or your account information to distinguish users that benefit from MySQL database co-tenancy. Again, it's possible to have multiple databases, owned by the same user or different MySQL users, running in a single MySQL database server instance. In the example shown in Figure 1-3, we used wp_ as a prefix for both usernames and database names, at least providing a hint to the database administrator that these belong to a WordPress installation.

What can go wrong between WordPress and MySQL? Three primary root causes of installation failure exist: your web server can't even find the MySQL server to begin with, it connects to the database but can't log in, or it logs in successfully but can't find the named database in which to create the WordPress tables. Note that all of these conditions need to be fulfilled at installation time; there has to be some basic database structure to contain the admin user before you can log in as that admin.

Web server can't find MySQL. Either you have the hostname for the MySQL server noted incorrectly in the `wp-config.php` file, or the web server is looking for a local MySQL instance and can't open the socket connection to it. Here's a simple example: When you run WordPress locally on MacOS, MySQL creates the socket `/tmp/mysql.sock` for local connections, but the WordPress PHP code is going to look for `/var/mysql/mysql.sock` through the PHP engine's MySQL module. Simply symbolically link one to the other:

```
# ln -s /tmp/mysql.sock /var/mysql/mysql.sock
```

The actual filesystem path to the local MySQL socket is a function of the database configuration; when it starts up it creates the local socket. Where the PHP engine, and therefore any PHP based applications, look for this socket is PHP configuration dependent. If you want to figure out exactly where the mismatch is, a bit of heavy-handed `printf()` style debugging helps.

Edit `wp-includes/wp-db.php`, the set of functions that establish WordPress's database connection. If you're seeing the "Error establishing a database connection" message during installation, insert an `echo(mysql_error());` statement where the error is detected to see the details displayed along with the generic message, as shown in Figure 1-5:

```
if (!$this->dbh) {
    echo(mysql_error());
    $this->bail(sprintf(/*WP_I18N_DB_CONN_ERROR*/"
    <h1>Error establishing a database connection</h1>
```

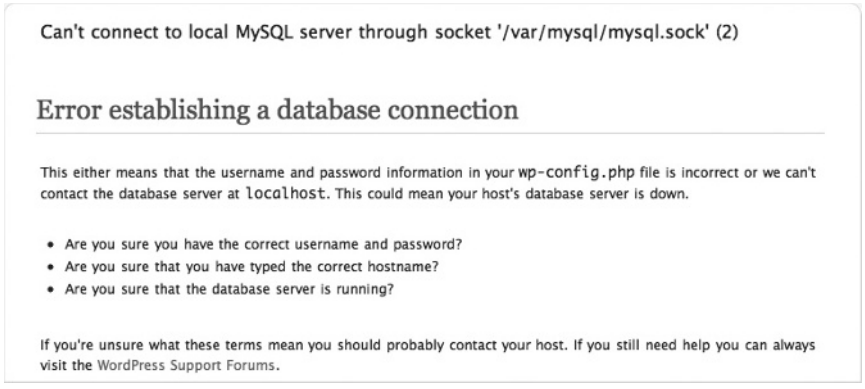
Can't connect to local MySQL server through socket '/var/mysql/mysql.sock' (2)
Error establishing a database connection
This either means that the username and password information in your wp-config.php file is incorrect or we can't contact the database server at localhost. This could mean your host's database server is down.
Are you sure you have the correct username and password?
Are you sure that you have typed the correct hostname?
Are you sure that the database server is running?
If you're unsure what these terms mean you should probably contact your host. If you still need help you can always visit the WordPress Support Forums.

FIGURE 1-5: mysql_error() reporting a socket problem

The `mysql_error()` function is a PHP library function that spits out the error generated by the last MySQL function called.

WordPress finds MySQL but can't log in. Most of the time, the MySQL username or password are wrong, particularly when you have to copy some arbitrary username generated by a hosting provider. Double-check your username data, and verify that it is reflected properly in your `wp-config.php` file. You may also run into a password authentication issue when using MySQL 4.1 or MySQL 5.0 with some web servers' PHP implementations; they only support the older MySQL 4.0 password hashing scheme. If this is the case, use MySQL's `OLD_PASSWORD()` function to hash your WordPress user's password in the backward-compatible format; the magic SQL incantation (at the MySQL command-line prompt or within the SQL window of MAMP) to address this is:

```
SET PASSWORD FOR user@host = OLD_PASSWORD('password');
```

where `user@host` is your WordPress database username and database hostname, and `password` is the (clear text) password you provided in the configuration file.

WordPress connects to MySQL but can't select the database. Just because the web server can log in to the database server with your WordPress database user information doesn't mean that there's necessarily a database available to that user. This is another scenario best diagnosed with `mysql_error()`, inserting it in `wp-db.php` where the selection error is identified:

```
function select($db) {
    if (!@mysql_select_db($db, $this->dbh)) {
        $this->ready = false;
        echo(mysql_error());
        $this->bail(sprintf(/*WP_I18N_DB_SELECT_DB*/'
        <h1>Can&#8217;t select database</h1>
        ..
```

If your attempts to complete installation result in an error box like that shown in Figure 1-6, after inserting the `mysql_error()` statement as described earlier, your MySQL database wasn't created under the appropriate database user, or the database user doesn't have privileges to use it. Double-check what MySQL believes using the command line:

```
% /usr/local/mysql/bin/mysql -u wp_user1 -p
Enter password:
Welcome to the MySQL monitor.  Commands end with; or \g.
Your MySQL connection id is 174
Server version: 5.1.37 MySQL Community Server (GPL)
mysql> show databases;
+ - - - - - - - - - - +
| Database           |
+ - - - - - - - - - - +
| information_schema |
| test               |
+ - - - - - - - - - - +
2 rows in set (0.00 sec)
```

Once we logged in as our designated MySQL database user, we didn't see the MySQL database — in this case, it was probably created by MySQL user root, and permissions to access or modify it weren't granted to the WordPress installation's MySQL user. . If you have MySQL root access, or sufficient

MySQL user privileges to create new databases within the MySQL instance, it's easy enough to create a database once logged in on the command line:

```
mysql> create database wp_halstern;
Query OK, 1 row affected (0.00 sec)
```

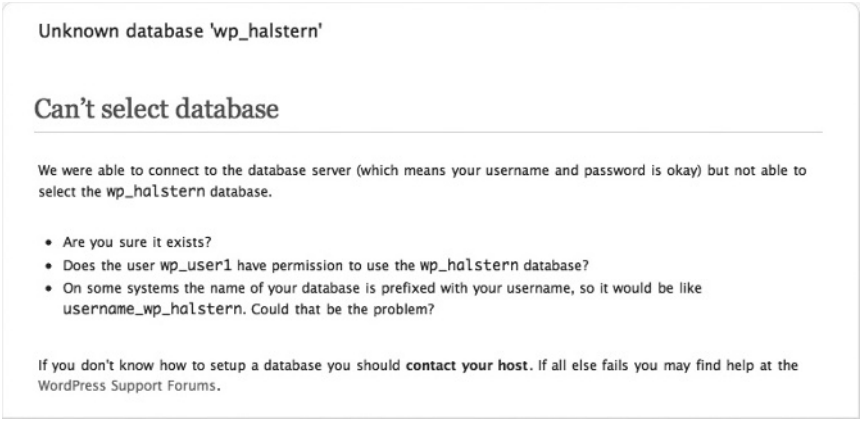
Unknown database 'wp_halstern'
Can't select database
We were able to connect to the database server (which means your username and password is okay) but not able to select the wp_halstern database.
Are you sure it exists?
Does the user wp_user1 have permission to use the wp_halstern database?
On some systems the name of your database is prefixed with your username, so it would be like username_wp_halstern. Could that be the problem?
If you don't know how to setup a database you should contact your host. If all else fails you may find help at the WordPress Support Forums.

FIGURE 1-6: MySQL database selection error

Again, it's important to distinguish operating system users from MySQL users from WordPress users. MySQL users are defined in the database and granted privileges to create databases, muck with tables, and otherwise generate useful data. WordPress users exist within the WordPress database tables created during install; they only have privilege, context, and meaning once you're logged in to WordPress.

Once you have a clean WordPress installation, you should see a collection of tables named according to the table prefix you set in `wp-config.php`; again, this is easy enough to verify using the MySQL command line:

```
mysql> use wp_halstern; show tables;
Database changed
+ – – – – – – – – – – – – – – – +
| Tables_in_wp_halstern    |
+ – – – – – – – – – – – – – – – +
| wp_hs_comments           |
| wp_hs_links              |
| wp_hs_options            |
| wp_hs_postmeta           |
| wp_hs_posts              |
| wp_hs_term_relationships |
| wp_hs_term_taxonomy      |
| wp_hs_terms              |
| wp_hs_usermeta           |
| wp_hs_users              |
+ – – – – – – – – – – – – – – – +
10 rows in set (0.00 sec)
```

In this example, we set the database table prefix to `wp_hs_`; if we later add another WordPress blog using the same database user and instance, we can simply set a different prefix and have the two blogs co-mingled in the same database table. We dig into the schema and uses of the ten basic WordPress database tables in Chapter 6. For now, once you are happily connected to MySQL, you're ready for some final clean-up and first-time blog administration.

FINISHING UP

Your MySQL database is up and running; there's a home for your content; and your web server is happily executing the WordPress core code.

First-Time Administration

Once you get the first-time admin user information shown in Figure 1-4, proceed to log in and you'll see the basic WordPress Dashboard captured in Figure 1-7.

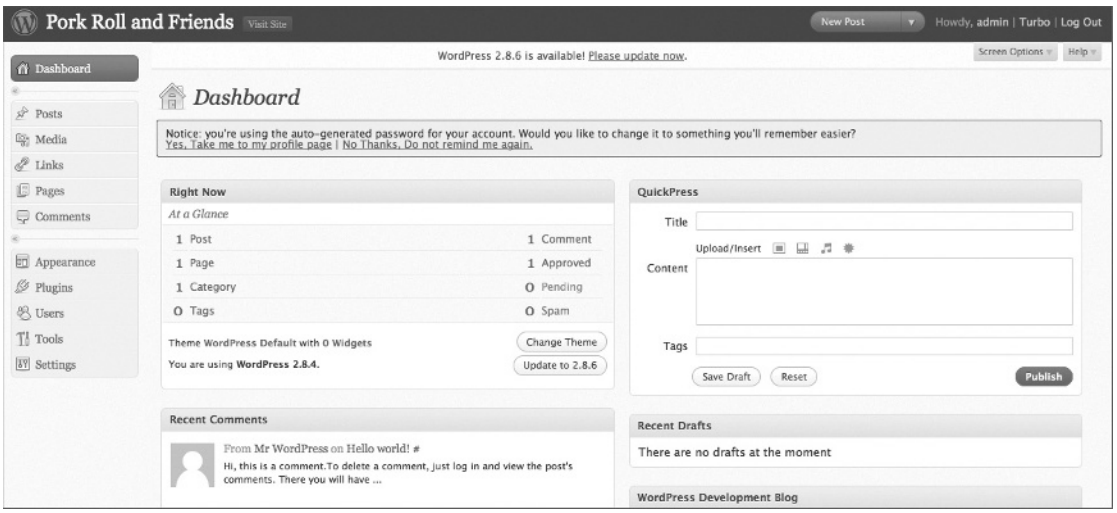

FIGURE 1-7: Dashboard view upon a first-time login

There's an admonition to change your default administrator password, which should be your first administrative step. If you're not redirected to the Dashboard through the Log In button, or if you happen to visit your blog's top-level URL first, either click the Log In link on your blog or explicitly go to the `wp-admin` subdirectory (`example.com/wp-admin`) to be presented with a login dialog box. Logging in to your blog takes you to the WordPress Dashboard, which is both amazingly simple in its power and rich in its complexity and exposed features.

What you do next with the Dashboard depends on how happy you are with the basic installation. If, as in the preceding example, you ended up with an older version of WordPress, click the Update to 2.8.6 button to do an in-place upgrade to the latest distribution. In addition to having a strong

self-installation feature, WordPress includes self-update functions (in `wp-admin/includes/update.php` if you're looking for them).

WordPress gently reminds you to change the administrative password upon your first login. You can also change the name of the admin user; if you intend to do most of the editing and control as the same user but want something more descriptive, change the username details in the Dashboard while you're setting up a new administrative password.

You may also decide to change some basic configuration options, such as the database name. or the MySQL database user, although you'll only change the default of `"root@localhost"` if you have full control over the web and database servers. The configuration file also has entries for "security keys" that are used to provide stronger security for browser cookies. Security keys are discussed in more detail in Chapter 11. Editing your `wp-config.php` file effects the changes right away. Changing the database table prefix, for example, causes WordPress to instantiate a new set of tables and create a clean-slate blog. Make those edits and then go back to your top-level blog URL and you'll find yourself with new admin user information and logged in to a starter Dashboard as in Figure 1-7. Old tables aren't removed from MySQL, so you'll have to do manual cleanup.

At this point, if you want to set your blog's URL to be different from the location in which you installed WordPress, you can choose Settings and General from the Dashboard and change the URLs for both your blog's top-level address as well as the WordPress installation directory. If you dissociate your blog's URL and the WordPress directory, make sure you move the `index.php` file to the desired top-level URL, and then edit the last line to include the proper subdirectory path to WordPress.

Before creating your first post, it's also a good idea to establish a permalink structure, so that everything you write follows the naming conventions you've chosen to make it relatively easy for readers to find, share, and link to your content. As expected, it's another option in the Settings portion of the Dashboard; options for permalink naming and their impact on performance and database schema are covered in more detail in the next chapter.

Whether it's really been five minutes, or a few hours of tracking down mismatches in hostnames, usernames, and database configurations, you're now ready to publish the first post of your own writing.

First Post

A successful WordPress installation already has a first post and comment published, both assuring that all of the moving pieces are moving in unison, and giving your blog some initial content. When you're ready to add your own first words, either use the right-hand QuickPress panel in the Dashboard to post an entry, or go to Posts and click Add New to be taken to the built-in WordPress editor. Figure 1-8 shows an entry in progress in the QuickPress panel, followed by the updated Dashboard after it's been successfully posted.

If your tastes run more old-school, you can always crank out content in your favorite text editor and then copy it into the editing pane. Be careful with WYSYIWIG word processors such as Microsoft Word or OpenOffice if you want to copy into the WordPress HTML composition window, because the HTML will be riddled with additional tag and style information. Finally, a variety of standalone blog editors, such as illumnix's ecto, publish to WordPress using the Atom Publishing Protocol or XML-RPC. Options for enabling posts to be published remotely are, as you'd expect, in the Dashboard's Settings section under Writing options.

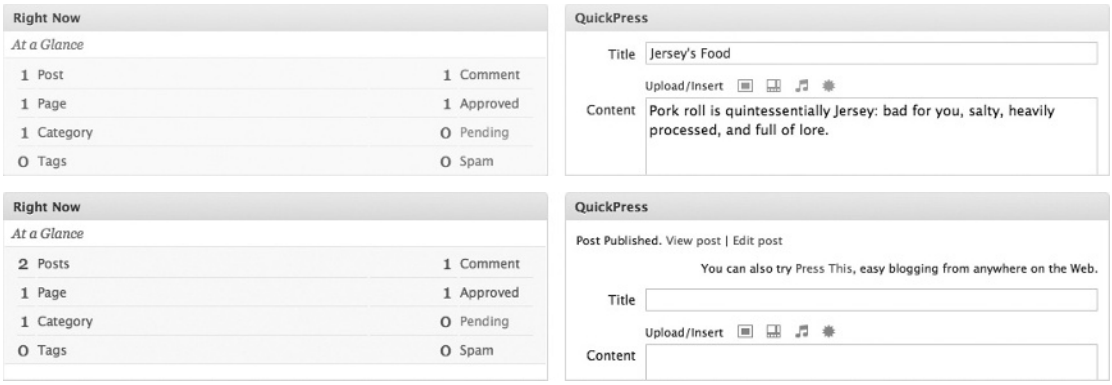

FIGURE 1-8: Publishing from the QuickPress panel

Click Publish for your own hello world moment. Multiple subsystems created that editing pane, saved the content in a database, generated and saved the referential metadata, and then emitted nice-looking HTML. Most of the user-visible pieces are governed through the Dashboard, and that's what we're going to cover next in this book, so you can change, improve, or integrate those systems to suit your content management and publishing needs. From there, we're going to dive into the core of WordPress so that you can take advantage of its extensibility, friendliness in design, and function.